The Comparison of Different Machine Learning Models for Predicting Default of Credit Card Clients

Group 6

Noah Olsen Jianing Wang Jia Chen

Professor Amir Jafari

DATS6202: Machine learning I

August, 2018

Outline

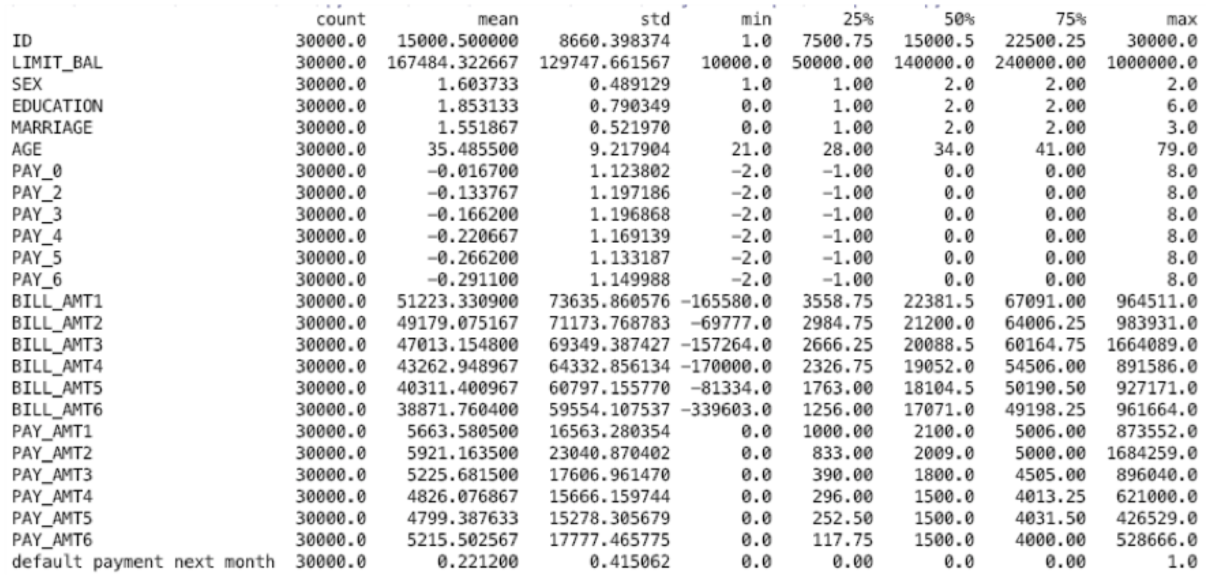
1. Introduction

As the use of credit cards becomes more convenient and offers better rewards than debit cards, credit cards became a more common method of payment in everyday use. But an impact of this method is, the more people use credit cards, the more problem arise. One of the problems is defaulting payment. If there was a method to predict the possibility of a consumer defaulting on payments, using computationally guided discovery, then this would save the company a tremendous amount of loss through proactive reporting.

This study aims to predict the possibility of default payments of credit card clients. To solve this problem, we plan to use four different approaches (Random Forest, Neural Network, Naïve Bayes Model, and Support Vector Machine) to forecast the default. Finally, the accuracy of those four methods will be compared.

2. Data Description

Our dataset is looking at defaults on credit card payments in Taiwan. Taken from the UCI Machine Learning Repository, the dataset has 30,000 observations and includes 23 explanatory variables along with a binary target variable in the last column on whether the person defaulted on the next month's payment (1 = Yes, 0 = No). Some brief descriptive info on the variables is presented in the table below using the pandas describe function.



```
                            count          mean           std       min       25%       50%        75%        max
ID                        30000.0   15000.500000    8660.398374       1.0   7500.75   15000.5   22500.25    30000.0
LIMIT_BAL                 30000.0  167484.322667  129747.661567   10000.0  50000.00  140000.0  240000.00  1000000.0
SEX                       30000.0       1.603733       0.489129       1.0      1.00       2.0       2.00        2.0
EDUCATION                 30000.0       1.853133       0.790349       0.0      1.00       2.0       2.00        6.0
MARRIAGE                  30000.0       1.551867       0.521970       0.0      1.00       2.0       2.00        3.0
AGE                       30000.0      35.485500       9.217904      21.0     28.00      34.0      41.00       79.0
PAY_0                     30000.0      -0.016700       1.123802      -2.0     -1.00       0.0       0.00        8.0
PAY_2                     30000.0      -0.133767       1.197186      -2.0     -1.00       0.0       0.00        8.0
PAY_3                     30000.0      -0.166200       1.196868      -2.0     -1.00       0.0       0.00        8.0
PAY_4                     30000.0      -0.220667       1.169139      -2.0     -1.00       0.0       0.00        8.0
PAY_5                     30000.0      -0.266200       1.133187      -2.0     -1.00       0.0       0.00        8.0
PAY_6                     30000.0      -0.291100       1.149988      -2.0     -1.00       0.0       0.00        8.0
BILL_AMT1                 30000.0   51223.330900   73635.860576 -165580.0   3558.75   22381.5   67091.00   964511.0
BILL_AMT2                 30000.0   49179.075167   71173.768783  -69777.0   2984.75   21200.0   64006.25   983931.0
BILL_AMT3                 30000.0   47013.154800   69349.387427 -157264.0   2666.25   20088.5   60164.75  1664089.0
BILL_AMT4                 30000.0   43262.948967   64332.856134 -170000.0   2326.75   19052.0   54506.00   891586.0
BILL_AMT5                 30000.0   40311.400967   60797.155770  -81334.0   1763.00   18104.5   50190.50   927171.0
BILL_AMT6                 30000.0   38871.760400   59554.107537 -339603.0   1256.00   17071.0   49198.25   961664.0
PAY_AMT1                  30000.0    5663.580500   16563.280354       0.0   1000.00    2100.0    5006.00   873552.0
PAY_AMT2                  30000.0    5921.163500   23040.870402       0.0    833.00    2009.0    5000.00  1684259.0
PAY_AMT3                  30000.0    5225.681500   17606.961470       0.0    390.00    1800.0    4505.00   896040.0
PAY_AMT4                  30000.0    4826.076867   15666.159744       0.0    296.00    1500.0    4013.25   621000.0
PAY_AMT5                  30000.0    4799.387633   15278.305679       0.0    252.50    1500.0    4031.50   426529.0
PAY_AMT6                  30000.0    5215.502567   17777.465775       0.0    117.75    1500.0    4000.00   528666.0
default payment next month 30000.0      0.221200       0.415062       0.0      0.00       0.0       0.00        1.0
```

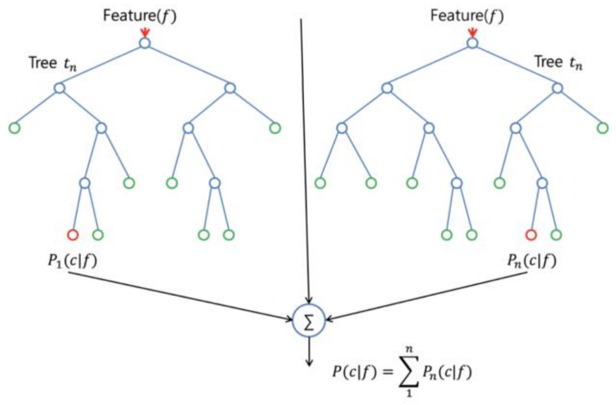
Column 1 is a unique person ID, column 2 is the person's balance limit, and columns of 3,4,5 and 6 are the person's sex, education level, marriage status, and age respectively. Columns 7-12 contain a history of past payments on a monthly level for the past 6 months coded based on what their payment status is like. Columns 13-18 have their monthly bill totals for the past 6 months. Columns 19-24 contain the dollar amounts they have repaid off of their bill on a monthly level for the past 6 months. After a brief EDA our plan is to test four different types of models to help predict whether or not the person is going to default on their bill.

We hope to test Random Forest, Neuron Network, Naïve Bayes Model, and Support Vector Machine. After looking at the misclassification rates for the four basic models we will choose 1 type of model to optimize for performance and try to minimize the misclassification rate.

3. Background Information on Learning Network and Algorithm

3.1 Random Forest

Random Forest is what is known as an ensemble model, taking many models and combining them into one. In the case of Random Forests, it is an ensemble of Decision Trees using the bagging method. A simple example of a Two-tree Random Forest is shown in the figure below



So what is happening is that the trees are calculated separately with different random combinations of features from the dataset and are then combined at the end to maximize performance using the bagging algorithm.

3.2 Neural Network

The type of Neural Network we are using in this project is a Multi-Layer Perceptron Neural Network (MLP). An MLP is a deep, artificial neural network composed of more than 1 perceptron or layer. The model contains an input layers that bring data features into the model, an output layer that makes a final prediction and then an arbitrary number of hidden layers in between the two that transform the data before making a final output decision.

3.3 Naïve Bayes

Naïve Bayes is one of classifiers in machine learning area basing on the Bayes' theorem. The aim of Naïve Bayes is to get the posterior probability, which is what kind of outcome basing on a series of given conditions. Figure 1 shows the basic equation of Naïve Bayes*.*

$$P(C|X) = \frac{[P(x_1|C) \times P(x_2|C) \cdots P(x_n|C)] \times P(C)}{P(X)}$$

Figure 1

First it calculates the posterior probability for each class variable by prior probabilities, frequency tables and likelihood tables. Then output is the class with the highest probability with conditions. During the entire process of Naïve Bayes, necessary calculations are some simple probability multiplication. Because of that, we can apply this algorithm faster than the other.

There are three different functions about Naïve Bayes we can find in Scikit-learn package: GaussianNB, BernoulliNB and MultinomialNB. The gaussianNB is often used for continuous variables. BernoulliNB is applicable to multiple binomial distributions, while MultinomialNB usually is for those discrete features.

3.4 Support Vector Machine

Support Vector Machines is one of supervised learning models. It can be used for classification and regression analysis. A support vector machine takes input data as vectors and outputs a hyperplane or set of hyperplanes in a high-dimensional space. The best hyperplane for Support Vector Machine is the one that maximizes the margins from both classes. For linear separable problems, there are 2 equations to find the optimal decision boundary as shown below:

- $\mathbf{w} = \sum \alpha_i y_i \mathbf{x}_i$

- $\sum \alpha_i y_i = 0$

For non-linear separable questions, kernel function is needed. In general, there are 3 used kernel functions: Linear kernel, Polynomial kernel, and the RBF kernel function. Linear kernel is simply defined by:

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x} \cdot \mathbf{x}'$$

Polynomial kernel:

$$K(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}' + c)^d$$

Sometimes polynomial kernels are not sophisticated enough to work. So we use RBF or Gaussian kernel:

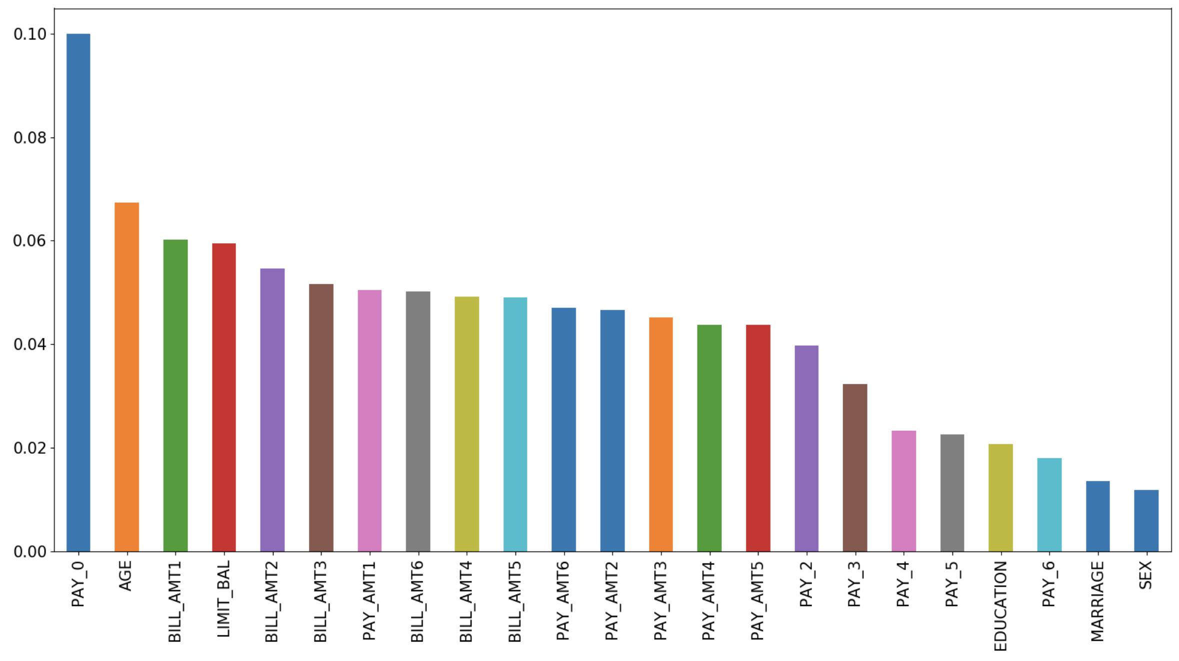
$$K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2)$$

Besides above kernels, there are still a lot of kernels available. Some of them are specific to a domain. (Kowalczyk, 2017)

4. Experimental Setup

4.1 Random Forest

For the Random Forest model we started by using sklearn RandomForestClassifier function. I started by looking at the feature importance, which yielded the plot below.



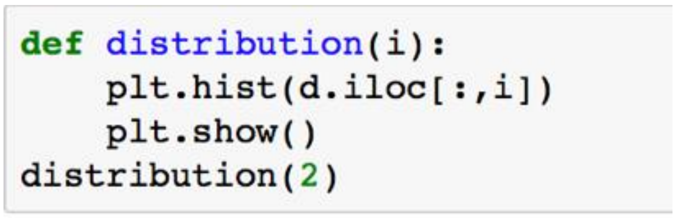
Given the fact that the dataset doesn't have that many features to begin with combined with the fact that many of the features appear to have similar importance I decided to use all of the features in the final model. I decided to use 100 classifiers as it performed the best without being too computationally intensive. I also tried using both a gini and entropy models. Other than that there are not too many tunable parameters in random forest models.

4.2 Neural Network

After having experimented with various experimental Neural Network setups using GridSearch with Keras and Stochastic Gradient Descent Algorithms I settled on using three hidden layers with 10 neurons in each layer with a max iteration value of 1000 before giving up on convergence of the model. Before running the model I normalized and standardized the dataset.

4.3 Naïve Bayes

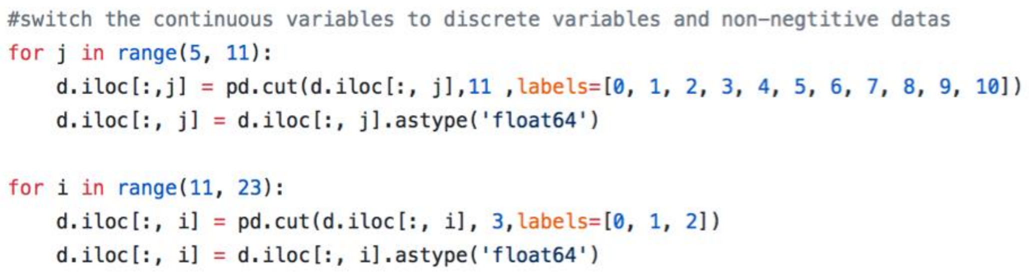
After reading in the dataset, the first thing is checking the type of the feature and class. Half of the features are discrete variables and the rest is continuous variables. Then we use function (Figure 2) to observe the distribution of the continuous variables. However, the last 12 columns distribute concentrated and almost have no normality trend. So, we decide to implement MultinomialNB function to train the data.



```
def distribution(i):
    plt.hist(d.iloc[:,i])
    plt.show()
distribution(2)
```

Figure 2

There are two prerequisites to apply MultinomialNB: discrete variables and non-negative variables. We transfer the continuous various to discrete ones and switch the negative data to non-negative. (Figure 3). We train the data after that, and the accuracy is only 58%. One of assumptions of Naïve Bayes is independence. If we can drop certain dependent variables, the accuracy of the model may go up. The next step is to check all the correlation (Figure 4) between each feature and delete some columns that are high correlated with each other. However, the result only increases by 0.1%.



```
#switch the continuous variables to discrete variables and non-negtitive datas
for j in range(5, 11):
    d.iloc[:,j] = pd.cut(d.iloc[:, j],11 ,labels=[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
    d.iloc[:, j] = d.iloc[:, j].astype('float64')

for i in range(11, 23):
    d.iloc[:, i] = pd.cut(d.iloc[:, i], 3,labels=[0, 1, 2])
    d.iloc[:, i] = d.iloc[:, i].astype('float64')
```

Figure 3

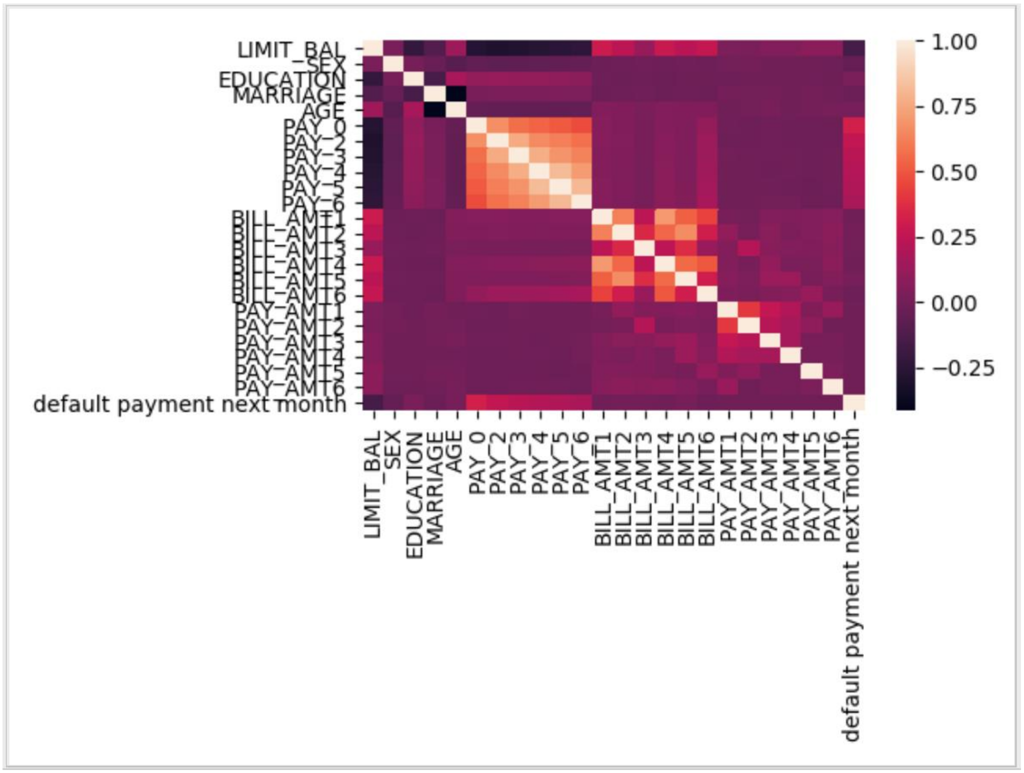
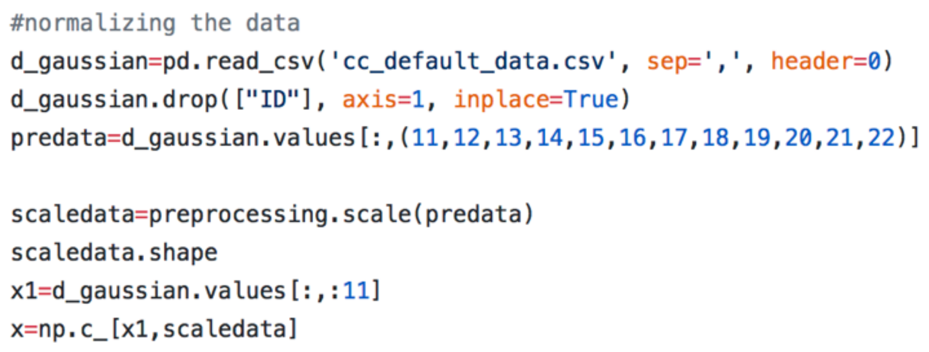


Figure 4

Because the accuracy is not good enough, so we want to try a different function. We can use BernoulliNB function only when every feature is binomial distribution. The last one we can experiment is GaussianNB. To apply GaussianNB function, the first thing to process data is normalizing (Figure 5). On one hand, it could decrease the probability of overfitting. On the other hand, after normalization, each feature has same scale to the outcome which is another assumption in Naïve Bayes and it can switch mean and variance to 0 and 1. The result rises up to 77.8% by using GaussianNB after it.



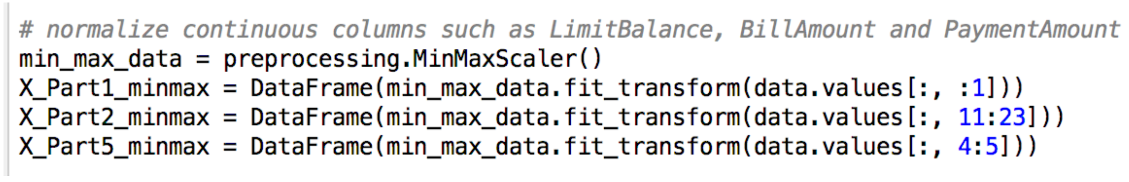
```
#normalizing the data
d_gaussian=pd.read_csv('cc_default_data.csv', sep=',', header=0)
d_gaussian.drop(["ID"], axis=1, inplace=True)
predata=d_gaussian.values[:,(11,12,13,14,15,16,17,18,19,20,21,22)]

scaledata=preprocessing.scale(predata)
scaledata.shape
x1=d_gaussian.values[:,:11]
x=np.c_[x1,scaledata]
```

Figure 5

4.4 SVM

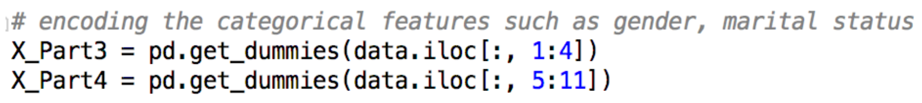
As a first step, let’s review the dataset. The dataset consists of 25 attributes. The first 24 columns contain information about ID, age, education, marital status, amount of given credit, repayment history of last 6 months (6 columns), amount of bill statement of last 6 column (6 columns), and amount of payment of last 6 months. The last column “default next month” is used as target. There are 2 values: 1 means default next month and 0 represents no default next month, so this is binary classification problem.

Next, it is important to decide which attributes to use and how to use them. In the beginning, name each column interpretably and briefly. It’s effective for showing the feature importance later. The data cleaning drops unnecessary columns and rows, such as “ID”, replaces missing data, and check the null values for each column. Though all of the data values are numeric, they can be divided into 2 types: continuous variables and discrete variables (categorical variables). For continuous variables, MinMaxScaler of sklearn.preprocessing package was used to normalize features to lie between 0 and 1. We tried MaxAbsScaler, scale, normalize as well, the result did not vary.



```
# normalize continuous columns such as LimitBalance, BillAmount and PaymentAmount
min_max_data = preprocessing.MinMaxScaler()
X_Part1_minmax = DataFrame(min_max_data.fit_transform(data.values[:, :1]))
X_Part2_minmax = DataFrame(min_max_data.fit_transform(data.values[:, 11:23]))
X_Part5_minmax = DataFrame(min_max_data.fit_transform(data.values[:, 4:5]))
```

For columns with categorical features such as marital status and gender, pandas.get_dummies was used to covert them into indicator variables.



```
# encoding the categorical features such as gender, marital status
X_Part3 = pd.get_dummies(data.iloc[:, 1:4])
X_Part4 = pd.get_dummies(data.iloc[:, 5:11])
```

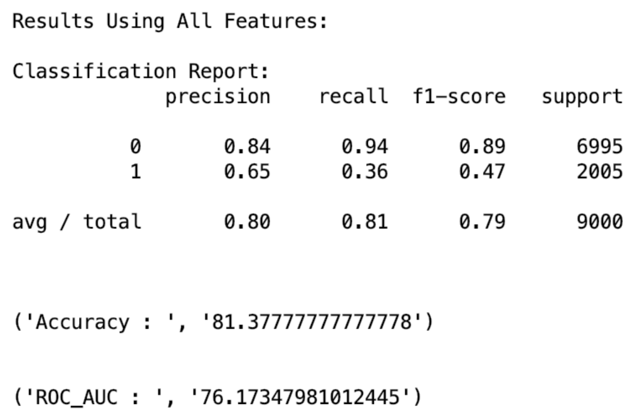
These parts were merged as our X_data. Then sklearn’s Label Encoder was used to encode our Y_data, thus our data were ready. Here we split the data into training and test data, respectively 70% and 30%.

For the choice of kernel function, it is recommended that try a RBF kernel first, because it usually works well. However, it is good to try the other types of kernels if you have enough time to do so (Kowalczyk, 2017). As suggested, we tried RBF at first, and obtained an accuracy was 80%. When changing the kernel to Linear, we obtained the same accuracy. For this data, both RBF and Linear worked equally well, so we selected the simpler one: Linear kernel.

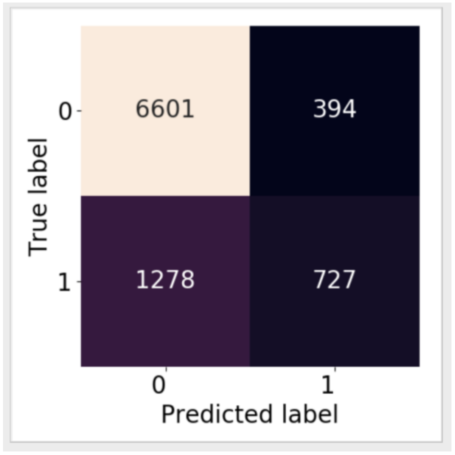
5. Result

5.1 Random Forest

The results for the random forest model were decent. The figures below show the classification report and confusion matrix for the model.



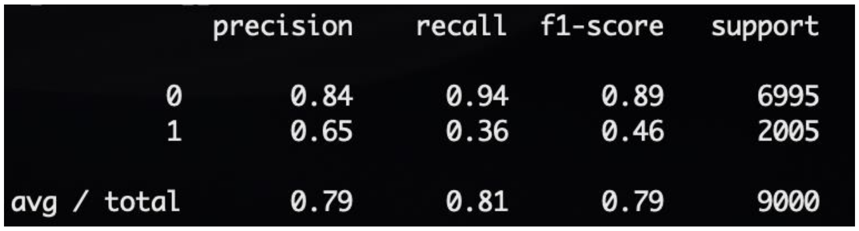
```
Results Using All Features:

Classification Report:
             precision    recall  f1-score   support

          0       0.84      0.94      0.89      6995
          1       0.65      0.36      0.47      2005

avg / total       0.80      0.81      0.79      9000


('Accuracy : ', '81.37777777777778')


('ROC_AUC : ', '76.17347981012445')
```



As can be seen, the model had an accuracy of 81.4%, but the confusion matrix shows that the accuracy isn't equal. While the mode is quite good at predicting when someone won't default on credit card payment (84% accuracy), the model isn't very good at predicting when someone will default (65% accuracy). Being as the goal is to predict when someone WILL default this might not necessarily be the most appropriate model even though the model sees decent overall performance.

5.2 Neural Network

The results from my best performing neural network are shown in the figure below.

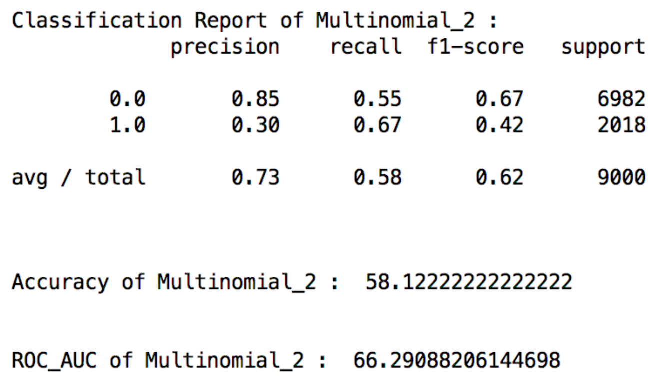


```
             precision    recall  f1-score   support

          0       0.84      0.94      0.89      6995
          1       0.65      0.36      0.46      2005

avg / total       0.79      0.81      0.79      9000
```

Similarly to the Random Forest Model, the Neural Network Model performed with about 80% accuracy at 79%. But also similarly to the Random Forest Model, the Neural Network was much better at predicting when someone won't default on their payment (84%) than when they will default (65%). Seeing as this model doesn't perform quite as well as the Random Forest Model it is still preferable to go with the Random Forest Model over this Neural Network Model.

5.3 Naïve Bayes

Figure 6 provides the classification report and confusion matrix of multinomial function. As can be seen in the percentage of recall, more test data that belongs to class 1 (default payment) have been predicted to the right class comparing to those in class 0. However, this is not a really good model as the accuracy is only 58.1%.



```
Classification Report of Multinomial_2 :
             precision    recall  f1-score   support

        0.0       0.85      0.55      0.67      6982
        1.0       0.30      0.67      0.42      2018

avg / total       0.73      0.58      0.62      9000


Accuracy of Multinomial_2 :  58.12222222222222


ROC_AUC of Multinomial_2 :  66.29088206144698
```

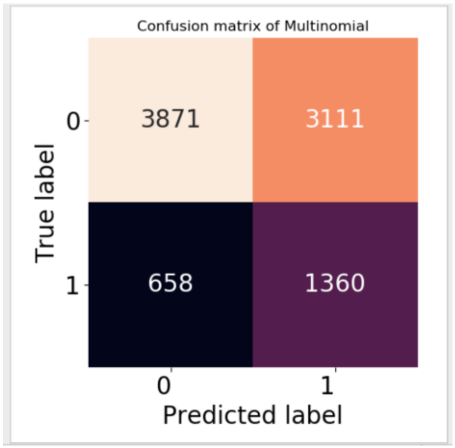
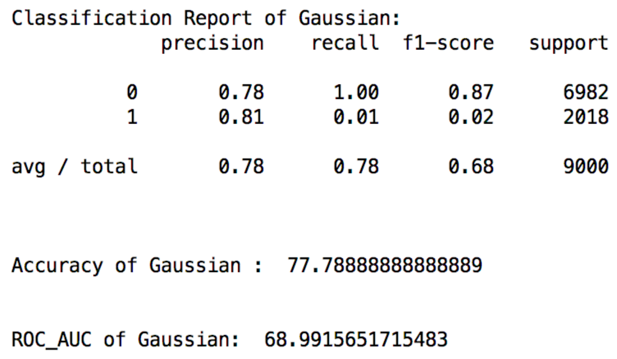


Figure 6

Confusion matrix and classification report of Gaussian is shown in Figure 7. According to Figure 7, the accuracy is about 20 percent higher (77.8%) than using multinomial function, and there are only 6 class 0 inputs in test data that have been classified to the 1 class.



```
Classification Report of Gaussian:
             precision    recall  f1-score   support

          0       0.78      1.00      0.87      6982
          1       0.81      0.01      0.02      2018

avg / total       0.78      0.78      0.68      9000


Accuracy of Gaussian :  77.78888888888889


ROC_AUC of Gaussian:  68.9915651715483
```

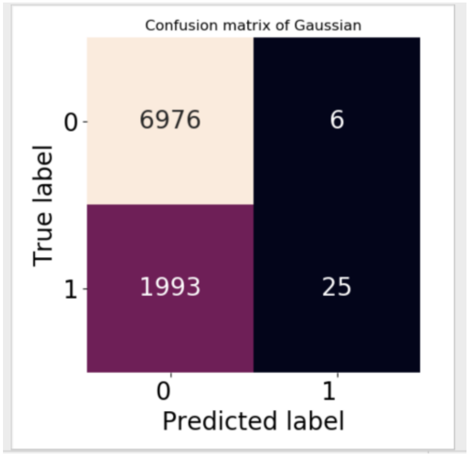


Figure 7

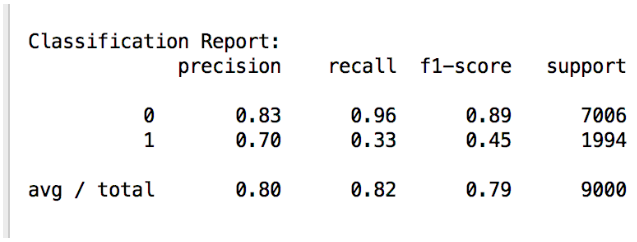
The objective of the project was to find a best model to predict whether the credit card user will have a default payment or not. Even though the accuracy in Gaussian function was higher than the other one, the recall for class I was only 1%, which means this model is not able to predict the default payment well.

This result may be explained by the limitation in Naïve Bayes that accuracy can be reduced if applied to large amounts of data or if the correlations among features are strong. Also, the types of the feature are various. It is hard to train if we only use the algorithm of Naïve Bayes before a bunch of complex data preparation work done. Therefore, Naïve Bayes is not an appropriate method to apply in this dataset.

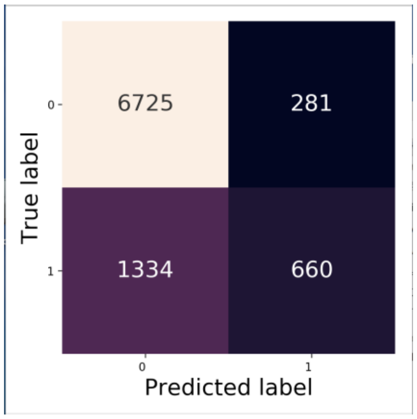
5.4 Support Vector Machine

The accuracy report and confusion matrix are shown below. According to these 2 figures, the accuracy of SVM model for this data is 80%. However, the precision of predicting class 1 is lower than class 0. It means that this model is stronger when predicting a client who won’t default payment to the class who won’t default. In contrast, it is weak to detect the client who will default. According to the confusion matrix, among the test data, there are 1994 clients who will default payment. However, this model only recalled 660 clients.

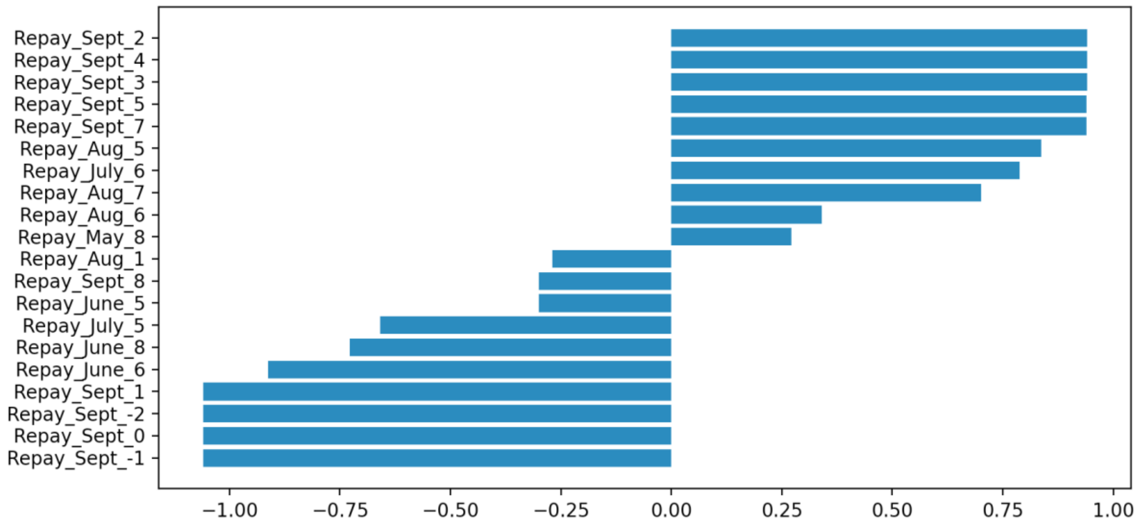
Considering the goal of bank is to reduce loss, this model should be improved the ability to predict default payment.



```
Classification Report:
                  precision    recall  f1-score   support

          0         0.83      0.96      0.89      7006
          1         0.70      0.33      0.45      1994

avg / total         0.80      0.82      0.79      9000
```

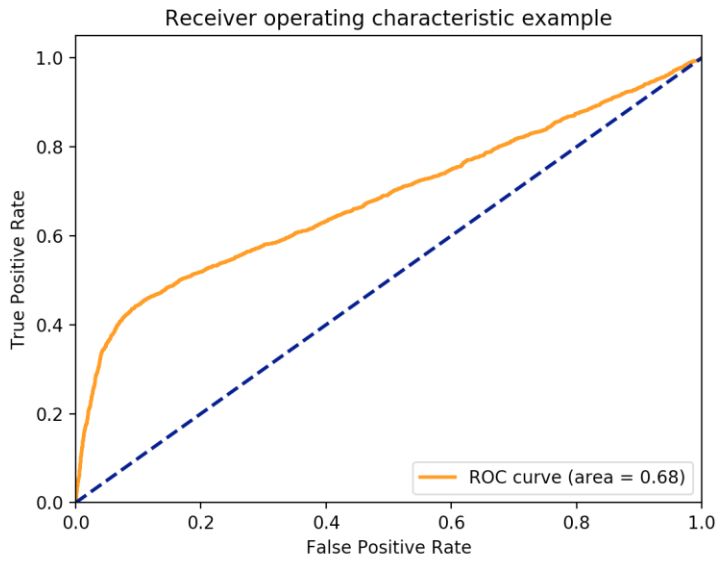


The figure above displays the top 10 positive and top 10 negative features.



First of all, take an example of the top positive feature Repay_Sept_2, it means client delay the payment for 2 months in September. -1 = pay duly; 1 = payment delay for one month; 2 = payment delay for two months; 8 = payment delay for eight months; 9 = payment delay for nine months and above. This model is aiming at predict whether client will default payment October. As can be seen, repayment history is predominantly important, especially the repayment status in September. It is concluded that if a client delay the repayment in September, it is prone to default payment in October. Otherwise, there is low probability for clients who repay duly to default payment in October.

The figure above shows the ROC curve and area. As can be seen, the ROC curve is in the above of blue dash line. It illustrates that this model is better than random classifier. However, it also demonstrates the fact that the accuracy of this model is not very good, just 80%.



6. Conclusion

This study analyzed the performance of four classifiers: Random Forest (RF), Neural Network (NN), Naïve Bayes (NB), and Support Vector Machine (SVM). In terms of accuracy, RF performed best, obtaining an accuracy of 81.4%. SVM (80.0%), NN (79.0%), NB (58.1% for multinomial function, 77.8% for Gaussian).

However, considering the precision of class 0 (won't default) and class 1 (default), it is noteworthy that all of those four classifiers performed better recognizing clients who won't default (0) than default (1). For the precision of default (1), SVM gained 70.0%, followed by RF 65.0%) and NN (65.0%). Although the precision of NB (Gaussian) was 81%, it only recalled 1% observation that belongs to class 1.

Above all, RF and SVM classifiers are good for this dataset. In contrast, NB is not an appropriate method for this dataset. For all of the models, the precision of predicting default needs to be enhanced.

7. Reference

Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480

Donges, N. (2018). The Random Forest Algorithm – Towards Data Science. Retrived From website: https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

Multilayer Perceptron. Retrived From website: *Skymind*, skymind.ai/wiki/multilayer-perceptron

Alexandre K. (2017). Support Vector Machines Succinctly. syncfusion